10-2006

Privacy Enhanced Superdistribution of Layered Content with Trusted Access Control

Daniel J. T. CHONG
Singapore Management University, danielchong@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1145/1179509.1179517

Privacy-Enhanced Superdistribution of Layered Content with Trusted Access Control

Daniel J. T. Chong
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
danielchong@smu.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management University
80 Stamford Road
Singapore 178902
robertdeng@smu.edu.sg

ABSTRACT

Traditional *superdistribution* [15] approaches do not address consumer privacy issues and also do not reliably prevent the malicious consumer from indiscriminately copying and redistributing the decryption keys or the decrypted content. The layered nature of common digital content can also be exploited to efficiently provide the consumer with choices over the quality of the content, allowing him/her to pay less for lower quality consumption and vice versa. This paper presents a system that superdistributes encrypted layered content and (1) allows the consumer to select a quality level at which to decrypt and consume the content; (2) prevents the merchant from knowing which exact content package is consumed by the consumer, hence enhancing consumer privacy; and (3) through trusted access control, prevents the consumer from indiscriminately copying and redistributing the decryption keys or the decrypted content, thus achieving a form of digital rights management.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General—*protection mechanisms*; K.4.1 [**Computers and Society**]: Public Policy Issues—*privacy*; K.4.4 [**Computers and Society**]: Electronic Commerce—*distributed commercial transactions, intellectual property, security*; D.4.6 [**Operating Systems**]: Security and Protection—*access controls*; K.5.1 [**Legal Aspects of Computing**]: Hardware/Software Protection—*copyrights, licensing, proprietary rights*

General Terms

Security, Design

Keywords

DRM, digital distribution, privacy, copyrights, usage rights, licensing, access control, trusted computing

DRM'06, October 30, 2006, Alexandria, Virginia, USA.

1. INTRODUCTION

The proliferation of affordable computing devices and broadband Internet access have led to increasing demand for multimedia digital content which include news, movies and music. Unlike traditional analog content which suffers from quality degradation when copied, digital content can be perfectly duplicated and then easily disseminated. Such piracy poses great difficulty in the enforcement of access rights and usage policies of the digital content. A market without reliable copyright protection will not last long, hurting both merchants and consumers in the long run. *Superdistribution* [15] is an effective solution to the above challenge. It encrypts the copyrighted content and distributes them via low-cost high-bandwidth distribution channels to all potential consumers of the information. Examples of such distribution channels include offline CD/DVD media, digital broadcasts and broadband Internet connections that allow for high-speed downloads as well as the increasingly popular peer-to-peer file transfers. As the copyrighted content is cryptographically protected in a *secure package*, to gain access to the content, the consumer will execute a *key acquisition protocol* between his/her device and a *key server* at merchant side. The amount of data exchanged in this protocol is in the range of only a few thousand bytes, so the consumer may use a separate low-bandwidth low-cost connection to acquire the key [13, 3]. To run a profitable business, the merchant will have to charge a fee for licensing its content to the consumer. According to a recent ACNielsen study [1], the most popular online payment methods are credit card, bank transfer, cash-on-delivery and PayPal, all of which require the consumer to divulge identification information to the merchant. This implies that in our key acquisition process, the consumer will have to submit his/her identification information[1], such as credit card number and/or email address. The release of any identification information to the merchant has implications on privacy.

Privacy has been a sensitive issue even before the advent of the Internet. However, the Internet creates many new threats to personal privacy and raises some unique privacy concerns [6, 17, 5]. Information sent over the Internet may

[1]This is assuming that the merchant requires the consumer to use a popular online payment method. As mentioned, all of the current popular online payment methods require the payer to provide identification information. There are of course payment methods which provide payer anonymity, such as the unsuccessful eCash by DigiCash, but these methods are unpopular and relatively inconvenient to use.

pass through dozens of different routers and computer systems on the way to its destination. Each of these systems may be capable of monitoring, capturing and storing online communications. The highly connected nature of the Internet makes it easy to automatically collect information from many different sources and compile a dossier on an individual - his/her likes and dislikes, shopping patterns, whereabouts and so on. Such data are a potentially valuable source of revenue for many businesses. Direct marketers can mine the data to derive targeted lists of users with similar preferences. The data can also be the source of abuses that may cause embarrassment to the users who have accessed sensitive or controversial materials online. While Internet users are understandably concerned about privacy when surfing the web, organizations are also taking customers' online privacy seriously to win their trust and to be compliant with privacy legislations.

A number of tools and systems have been developed to provide the Internet user anonymity while retrieving information over the Internet. *Onion Routing* [19] and *Crowds* [20] are two systems that uphold user anonymity in the Internet. Onion Routing is a general purpose infrastructure for anonymous communication over a public network. Crowds is a system for protecting user anonymity on the web based on the concept of "blending into a crowd" and operates by grouping users into a large and geographically diverse group, or crowd, that collectively issues HTTP requests on behalf of its members. In Crowds, web servers are unable to trace the source of a request because it is equally likely to have originated from any member of the crowd, and even collaborating crowd members can not distinguish the originator of a request from a member who is merely forwarding the request on behalf of another. The above anonymous systems are useful for web surfing in which users are not required to be identified and have no desire to do so. They are mostly useful when users visit web sites and download free digital products. However, in a superdistribution system, the consumer has to submit to the key server his/her identification or authentication information (e.g. credit card numbers or membership account information) in exchange for a cryptographic key to unlock the content in the secure package. Therefore, with respect to the issue of privacy protection in superdistribution, our focus is not on consumer anonymity, but on how to hide his/her shopping patterns as much as possible from the key server and other data collectors. This problem is in essence orthogonal to the anonymous communications problem.

Many digital content are organized as ordered layers, with each layer being a quality increment over the set of its preceding layers. In other words, each additional layer further refines the combined quality of its preceding layers. Examples of such layered content include JPEG2000 images, MPEG videos, structured literature such as news articles which present the most important information first followed by additional but lower priority information, and to a certain extent, even computer games with different levels of play.

Bao *et al.* proposed in [2] two protocols that prevent the merchant from finding out which specific digital product the consumer is purchasing. Zhu *et al.* proposed in [25] a key scheme for layered access control of MPEG-4 FGS videos, based on a cryptographic hash function and the Diffie-Hellman key agreement.

In this paper, we extend the techniques given in [2, 25] and propose a system and the corresponding protocols to protect consumers' privacy in the superdistribution of general layered content and to prevent a malicious consumer from pirating the decrypted content. The proposed system allows a consumer to disclose his/her identity information to a key server in exchange for a digital product (or equivalently, the cryptographic key to unlock the digital product), but prohibits the key server from learning which specific product the consumer obtains. With trusted access control, the system prevents the consumer from copying and illegally redistributing the unlocked digital product. Contributions of this paper include (1) a technique for encrypting layered content such that the consumer only needs to acquire a *single key* from the server in order to consume a subset of the layers; (2) a *privacy-enhancing* technique to prevent the key server from knowing exactly which product the consumer is purchasing, by logically grouping products of similar values and applying a commutative cipher; and (3) extending a *trusted virtual machine monitor* (TVMM) with a shared disk and applying it at consumer side to achieve trusted access control.

In our system, to prevent a malicious consumer from pirating the digital content and to achieve trusted access control, the consumer is required to run a *reader application* (which may also be called the *DRM client*), e.g. image viewer, within a closed-box *virtual machine* (VM) that is monitored by a *trusted virtual machine monitor* (TVMM) such as Terra [11]. Terra is a **hardware-level** *virtual machine monitor* (VMM) that multiplexes multiple VMs on a single hardware platform, as opposed to a **paravirtualizing** VMM which requires the hosted operating system (OS) to be ported or a **binary translator** VMM which translates instructions during runtime [11, 21]. Terra exposes to each VM a virtual hardware interface which is identical to that of the underlying physical hardware. Thus, an OS running in the VM accesses the exposed virtual hardware the same way it would access the actual physical hardware. VMMs are not new. They have been heavily researched in the 1960s, but were obviated in the 1980s when multitasking operating systems became popular. However, research interest in VMMs has recently been rekindled, with the focus shifted from merely multiplexing hardware to ensuring security and reliability in software systems [21]. Both Intel and AMD have announced plans to release their hardware-level virtualization technologies targeted at both enterprise servers and consumer desktops [7]. A TVMM such as Terra runs on commodity hardware and exports two VM abstractions: *open-box* and *closed-box*. An open-box VM provides the semantics of an open platform and can host a regular commodity OS running commodity applications; a closed-box VM provides secure isolation from other VMs and can host a thin trusted OS running a trusted application. The TVMM provides efficient and secure isolation between the VMs. Not even the platform owner can inspect or manipulate the contents of a closed-box VM [11]. It is in such a closed-box VM, extended with access to a *shared disk*, that our reader application runs. The system we propose is efficient in its operations and do not impose unacceptable processing burdens on both the consumer and the key server.

The rest of the paper is organized as follows. In Section 2 we present a basic superdistribution system for distributing

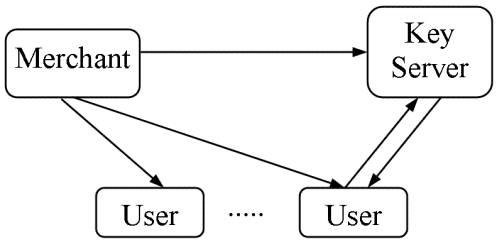

Figure 1: A typical superdistribution system model.

$M_0 = \{L_0\}$ $M_1 = \{L_0|L_1\}$ $M_2 = \{L_0|L_1|L_2\}$

Figure 2: A JPEG2000 code stream displayed in 3 qualities.

scalable multimedia content. In Section 3 we describe the privacy enhanced version of the system given in Section 2. In Section 4 we further enhance the system with trusted access control. Section 5 discusses some practical operation issues of the proposed system and Section 6 concludes the paper.

2. BASIC SUPERDISTRIBUTION OF MULTIMEDIA CONTENT

Superdistribution applies to all types of digital content, such as multimedia, data and software. With multimedia content accounting for a significant portion of all Internet traffic, we hereby focus our discussion on such content without loss of generality. The superdistribution system model under consideration is shown in Figure 1.

The system operation consists of the following high-level steps:

1. Generation of Secure Package: A content producer or authorized merchant encrypts a digital product, and places the resulting ciphertext in a secure package.

2. Distribution of Secure Package: The merchant distributes the package to the consumers. Since the digital product is protected by encryption, all conceivable channels of content distribution can now be used, including digital cable TV, satellite broadcast, CD/DVD-ROM publishing, and of course general downloading as well as P2P file transfer over the Internet. The keys for decrypting the secure packages are forwarded securely from the merchant to an on-line key server, which serves as an agent for multiple content providers.

3. Key Acquisition: A consumer who desires to access a digital product sends his/her request together with authentication information to the key server, which responds with the appropriate decryption keys according to consumer's privilege or amount of payment.

In order to achieve scalable and flexible access control to encrypted content, generation of the secure packages should take the data structure of the protected content into consideration. We illustrate this point using JPEG2000 [22, 18] as an example. JPEG2000 is an emerging ISO standard for still image compression designed to address most of the limitations of the original JPEG standard. It has a remarkable "*compress once, decompress many ways*" property, in that it supports extraction of transcoded images with various resolutions, quality layers and regions-of-interest, all from the same compressed *image code stream*. Exploiting this property, applications are able to disclose only the required image data (known as *transcode* in JPEG2000) of a particular code stream for any target consumer based on his/her privileges or capabilities.

Data in a JPEG2000 code stream can be arranged in *quality layers*. As an illustration, consider a code stream with three layers $M = \{L_0, L_1, L_2\}$, where L_0 is the base layer, and L_1 is the first enhancement layer and L_2 is the second enhancement layer. Let $x|y$ denote the concatenation of x and y. The transcoded images of basic quality, moderate quality and high quality are represented respectively by $M_0 = \{L_0\}$, $M_1 = \{L_0|L_1\}$ and $M_2 = \{L_0|L_1|L_2\}$, as illustrated in Figure 2.

We denote a general JPEG2000 code stream as $M = \{L_0, L_1, \ldots, L_I\}$ and denote its transcoded image of i^{th} quality as $M_i = \{L_0|L_1|\ldots|L_i\}, i = 0, 1, \ldots, I$. To produce a secure package for code stream M, the merchant proceeds as follows:

Generation of Secure Package

1. Choose a random key k_I and compute $k_{i-1} = h(k_i)$, $for i = I, I-1, \ldots, 2, 1$, where $h()$ is a one-way hash function. Hence, there are a total of $I+1$ keys. However, only k_I needs to be stored in the key server.

2. Encrypt each L_i with key k_i to obtain ciphertext $C_i = e(k_i, L_i), i = 0, 1, \ldots, I$, where $e(x, y)$ is a symmetric key encryption of message y using key x. An example of such an encryption algorithm is AES [9].

3. Construct the secure package of code stream M as $< B, \{C_i; i = 0, 1, \ldots, I\} >$, where B is the metadata of the code stream which contains information such as the code stream identifier, title, producer, number of quality layers, and usage rules.

The application of the hash function chain in Step 1 reduces the key management overhead, which will be discussed shortly. The secure package constructed in Step 3 is to be distributed freely to all the potential consumers.

Now, suppose that a consumer wishes to access the i^{th}-quality transcoded image of the code stream. He/she will execute the following key acquisition protocol with the key server to acquire the corresponding key. In the description of the key acquisition protocol, and henceforth, we use the term "consumer" interchangeably with the term "reader application".

Key Acquisition

1. The consumer sends his/her authentication data, the code stream identifier (which the consumer obtains from B in the secure package) and the requested the quality (i.e. the i^{th} quality) to the key server. At this

point, the key server authenticates the consumer or requires the consumer to make payment.

2. The key server, using the code stream identifier as index, retrieves the corresponding k_I from its database. The key server then computes $k_{I-1} = h(k_I), k_{I-2} = h(k_{I-1}), \ldots, k_i = h(k_{i+1})$, and sends k_i to the consumer.

3. Upon receiving k_i, the consumer computes $k_{i-1} = h(k_i), k_{i-2} = h(k_{i-1}), \ldots, k_0 = h(k_1)$. He/she then uses the keys $k_0, k_1, \ldots, k_i$ to respectively decrypt C_0, $C_1, \ldots, C_i$ to get the desired transcoded image $M_i = \{L_0|L_1|\ldots|L_i\}$.

It is noteworthy that our secure package is constructed such that consumers can obtain multiple transcoded images from the single package. This design fully preserves JPEG2000's "compress once, decompress many ways" property. It is also interesting to note that, due to the application of hash chaining, we require only *one* key to be sent from the key server to the consumer, regardless of the quality of the requested transcoded image. Doing so reduces both the storage requirement for storing keys and the amount of data transmitted. The reductions are especially significant when the number of quality layers is large. As a side note, JPEG2000 supports up to 65,535 quality layers in a single code stream. We also note that the above technique can be extended to video streams, such as Motion JPEG2000 [10] and MPEG4 streams [14], as well as other digital products with layered content.

3. PRIVACY ENHANCED SUPER-DISTRIBUTION

In this section we present a privacy enhanced superdistribution system which (a) allows a consumer to disclose his/her identity information (such as user account information or credit card number) to the key server in exchange for a decryption key of a digital product; (b) prevents the key server from learning which specific decryption key (and hence identifying the product) the consumer intends to obtain; and (c) prevents the consumer from obtaining more than what he/she deserves to obtain.

The proposed system is an extension of the technique given in [2], which is designed for generic content. Our extension deals with the efficient superdistribution of multilayered content. Our system employs the following commutative cipher known as the Pohlig-Hellman exponentiation cipher [16]. Let p be a prime number such that computation of discrete logarithm modulo p is infeasible, $a, 0 < a < p-1$, a random number which is coprime to $p-1$, and $b = \frac{1}{a} \bmod (p-1)$. Let $X, 0 < X < p$, be a message. Then the encryption of X using the encryption key a is $D = X^a \bmod p$ and the decryption of D using the decryption key b is $X = D^b \bmod p$. It is easy to see that this cipher is commutative.

Assume that the content merchant has many JPEG2000 code streams, each supporting $I+1$ transcoded images of different qualities. The merchant first sets up $I+1$ commutative ciphers. Let p_i be a prime, a_i, co-prime to $p_i - 1$ and $0 < a_i < p_i - 1$, be an encryption key and $b_i = \frac{1}{a_i} \bmod (p_i - 1)$ be the corresponding decryption key, $i = 0, 1, \ldots, I$. We will use the same a_i and b_i to respectively encrypt and decrypt the i^{th} layer of every code stream. The merchant operates as follows to generate a secure package for a particular JPEG2000 code stream $M = \{L_0, L_1, \ldots, L_I\}$. As in the previous section, we use $e(x, y)$ to denote the encryption of message y using key x with a symmetric key cipher.

Generation of Secure Package

1. Choose a random key k_I and compute $k_{i-1} = h(k_i)$, for $i = I, I-1, \ldots, 2, 1$.

2. Encrypt L_i with key k_i to obtain $C_i = e(k_i, L_i)$, and encrypt k_i with key a_i to obtain $D_i = (k_i)^{a_i} \bmod p_i, i = 0, 1, \ldots, I-1, I$. Note that even for another code stream, with obviously a different key k_i, the same a_i is used in encrypting that key.

3. Construct the secure package of the code stream as $< B, \{C_i, p_i, D_i; i = 0, 1, 2, \ldots, I\} >$, where B is the metadata of the code stream.

All the secure packages are distributed freely to the consumers. Assuming that a consumer wants to access a particular transcoded image M_i (of i^{th} quality) from the secure package of M, he/she will execute the following protocol with the key server to acquire the corresponding key:

Key Acquisition

1. The consumer randomly picks a number r, where $0 < r < p_i - 1$, computes $s = \frac{1}{r} \bmod (p_i - 1)$ and $U = (D_i)^s \bmod p_i$. He/she then sends U, his/her authentication data and the requested quality (i.e., the value i for the i^{th} quality) to the key server. Note that the code stream identifier does not need to be sent. At this point, the key server can require the consumer to make payment.

2. The key server computes $V = U^{b_i} \bmod p_i$ and returns V to the consumer.

3. The consumer recovers the key $k_i = V^r \bmod p_i$, computes k_j recursively from k_i using the hash function $h(), j = i-1, \ldots, 1, 0$, and decrypts $C_j, j = 0, 1, \ldots, i$, to obtain the desired transcoded image $M_i = \{L_0|L_1|\ldots|L_i\}$.

Note that $U = (D_i)^s \bmod p_i = ((k_i)^{a_i})^s \bmod p_i, V = U^{b_i} \bmod p_i = (k_i)^s \bmod p_i$; therefore $V^r \bmod p_i = k_i$ is the key desired by the consumer. This protocol achieves information-theoretical security for consumers. The key server only knows U and is not able to derive D_i, and thus unable to know which decryption key is requested by the consumer. Hence, the key server only knows that the consumer requested a code stream of a particular quality, but does not know which particular code stream the consumer has requested. This achieves our design objective on protecting consumer privacy.

A practical consideration for the merchant is that the secure packages, which are distributed as a logical group, should each have a similar financial value. As there is no way for the merchant to know which package the consumer is using, the only practical measure is for the merchant to fix the same price for each i^{th}-quality layer across all the packages. This constraint should not be too limiting if there are sufficient packages of similar values.

4. ACHIEVING TRUSTED ACCESS CONTROL

Thus far, we have described a privacy-enhanced superdistribution system that prevents the merchant from knowing what the consumer purchases, and has the flexibility of allowing the consumer to select a quality level at which to consume the content and pay in accordance to the selected quality, both without imposing impractical implementation burdens. However, a malicious consumer may break the local software to obtain k_i and pass the key to others. In this section, to prevent software tampering, we further enhance the system to include trusted access control at the consumer side.

To achieve trusted access control, we require the consumer to run a *trusted virtual machine monitor* (TVMM) such as Terra [11] on commodity PC hardware equipped with a *Trusted Platform Module* (TPM) [23]. Many new commodity desktop PCs and laptops have already been equipped with TPMs, including those from Dell, Fujitsu, HP, Toshiba and Lenovo [24]. IDC predicts that by 2010 TPM adoption will increase to near ubiquity for both business and consumer PC hardware [8]. Based on open industry specification, the TPM has two components: a secure cryptographic microcontroller similar to a smartcard microcontroller and a software interface for TPM-aware applications. An important aspect of the TPM is that it is the key enabler for *remote attestation*. With remote attestation, an application can authenticate itself, as well as the OS it is running on and the hardware platform hosting the OS, to a remote party. To perform the authentication, the application requests the OS for an endorsement. The OS in turn requests the lower layer, which could be a TVMM or a boot loader, for an endorsement. This goes on until the TPM-equipped hardware platform at the lowest layer is reached. Then, starting at the lowest layer, each layer signs a hash of the upper layer, building a hash chain all the way up to the application. The entire certificate chain is sent to the remote party, which subsequently verifies each certificate and checks that the hash value of each layer corresponds to the known trusted value stored at the remote party side. When verified, the application in question is trusted to behave as expected by the remote party. With *direct anonymous attestation* [4], the verifier only learns that the consumer uses a TPM but not which particular one. This is a good-to-have property, but not important in our system.

The TVMM exports an open-box and a closed-box VM abstractions. The consumer can continue to run a commodity OS such as Linux or Microsoft Windows within an open-box VM. This VM exposes the same semantics as the underlying general-purpose hardware platform, such that the OS and the installed applications can run as though they were running right on top of the hardware platform. The consumer will use this open-box VM for his normal computing activities, which can certainly include using the web browser and tools such as P2P file sharing software to download encrypted packages from the merchant. He/she may also run multiple instances of open-box VMs, but the TVMM will ensure that each VM is run in isolation and each uses a dedicated virtual hard drive independently for storage. Within a closed-box VM, a thin trusted OS is hosted and the trusted reader application installed. The reader application enforces the usage policy dictated by the merchant, which may include how many times the consumer can consume the content and when he/she can do so. Because the main purpose of the thin OS is to run only the single reader application, it does not require multitasking capability and hence can have a small footprint. Figure 3 illustrates how the VMs run within the TVMM.

In addition to providing the same hardware interface as open-box VMs, the closed-box VM has access to a narrow interface provided by the TVMM for performing attestation [11]. Further to that, we provide a separate narrow interface with file IO functions for all the VMs to access a *shared disk*. In a traditional TVMM such as Terra, inter-process communication (IPC) between VMs is not supported, other than by emulating a network [12]. This means that there is previously no way for VMs to communicate or share data intrinsically. With the introduction of a shared disk, we allow the consumer to download the secure package files into the shared disk from within the open-box VM environment, so that the files can be accessed by the reader application in the closed-box VM later. The TVMM will ensure that the VMs access the shared disk in a mutually exclusive manner.

The following are the steps that the consumer will go through to consume the superdistributed content with trusted access control:

1. Download Secure Package: In the open-box VM, the consumer uses a convenient method to obtain the superdistributed secure package. For example, he/she could do so through downloading from a website. The package file is stored in the shared disk.

2. Perform remote attestation: From within the open-box VM, the consumer will perform a context switch to the closed-box VM. There, he/she will run the reader application, which will initiate an SSL handshaking process to authenticate the merchant key server and to establish a session key for secure communication. Remote attestation is subsequently executed with the server to authenticate the entire software stack at consumer side, from hardware and firmware up to the boot loader, TVMM and the reader application. Through the narrow attestation API provided by the TVMM, the reader application will call an "Endorse" function which triggers a series of attestation-related events, as mentioned in detail earlier in this section. The resulting hash chain is then presented to the server. The server has to verify all the certificates and hash values, before trusting that the reader application will behave as expected.

3. Key acquisition: The consumer will now execute the key acquisition protocol with the key server, as described in detail in the previous section, except that now the messages are exchanged between the trusted reader application and the key server through the established secure SSL communication channel. It is important that key acquisition process be protected by the SSL session. Otherwise, the key acquisition process would be susceptible to man-in-the-middle attack, in that immediately after attestation is completed, a malicious consumer could intercept the insecure communication channel and replace the reader application with a misbehaving one running on another machine, hence bypassing the merchant's usage policy. At the

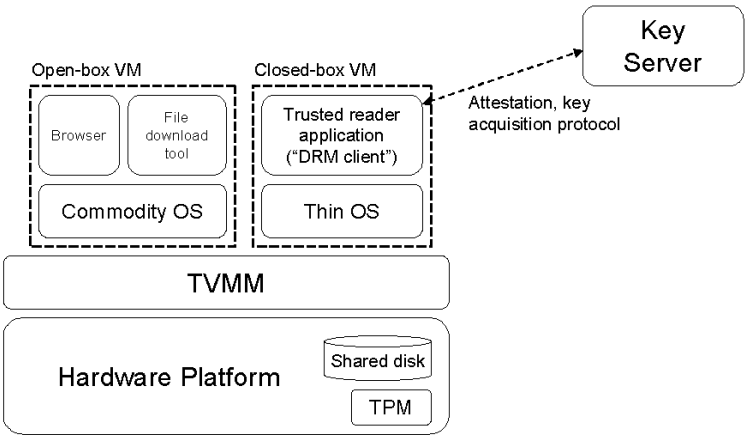

Figure 3: The open-box VM and closed-box VM running on top of the TVMM, both accessing the *shared disk*.

end of this key acquisition protocol, the reader application will have decrypted the package retrieved from the shared disk and obtained the user-desired transcoded image.

4. Consume package: The reader application will allow the consumer to view the transcoded image in accordance to the usage policy dictated by the merchant. The simplest usage policy will require the consumer to always perform attestation and key acquisition (which may include payment) whenever he/she wants to consume the package. In this case, access control data is not stored at consumer side. Alternatively, during key acquisition, the key server can send a policy file together with the decryption key to the reader application. The policy file can contain access control information such as the number of times the consumer can access the decrypted data. This file will be stored by the TPM in sealed storage, which is accessible only when the reader application has been attested. The consumer can run the same reader application to consume the data without having to perform attestation and key acquisition, as long as the usage policy allows so.

Successful attestation ensures that the consumer is running a reader application that is trusted by the merchant to behave in accordance to the merchant's specification. It also ensures that the thin OS on which the reader application runs is also trusted not to misbehave, such as not dynamically modifying the reader application. The TVMM, boot loader, firmware and hardware are all similarly trusted to behave as expected. Hence, the merchant can be sure that the decryption keys as well as the decrypted content do not leave the trusted environment. That is, the keys and content will not be copied and distributed to a third party.

5. PRACTICAL CONSIDERATIONS

Communication and Computational Overhead: First we consider communication overhead. In the original superdistribution system, the merchant sends $\{B, k_I\}$ to the key server for every digital product while in the privacy enhanced system only $\{b_i, i = 0, 1, \ldots, I\}$ are sent from the merchant to the key server. The amount of overhead in the former is in proportional to the number of products while that in the latter is fixed. In the privacy enhanced system, additional overhead to convey keying information, $\{p_i, D_i; i = 0, 1, \ldots, I\}$, is introduced in a secure package. However, this overhead is not an important concern since it is distributed over high bandwidth and low cost channels. Next, we compare computational overhead between the two systems. To answer a consumer request, the key server in the original system carries out a hash chain computation while that in the privacy enhanced system performs a decryption operation of the commutative cipher. Both operations are considered low cost for a server machine. In the trusted version of the system, the additional attestation and SSL communication requirements also do not impose a heavy computational burden on the server.

Sales Statistics Collection: We note that in practice it is often necessary for content providers or clearing houses to know the sales statistics of particular products, such as the number of copes sold in a day. Fortunately, it is possible to achieve both user privacy protection and sales statistics collection at the same time [2].

6. CONCLUSION

In this paper we have proposed a system and the corresponding protocols for privacy enhanced superdistribution with trusted access control. We presented our system using multimedia content as an illustrative example, but the technique applies to all layered content types. Privacy protection gives users peace of mind. Hence, everything else being equal, content providers who respect user privacy are more attractive to users than those who do not.

7. ACKNOWLEDGEMENT

This paper is funded by the Office of Research, Singapore Management University.

8. REFERENCES

[1] ACNielsen. One-tenth of the world's population shopping online. http://www2.acnielsen.com/news/20051019.shtml, October 2005.

[2] F. Bao and R. H. Deng. Protocols that hide user's preferences in electronic transactions. *Journal of Computer Networks*, 48(4):503–515, December 2004.

[3] P. Benton. Packing information for superdistribution. http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/packinginfo396.html, 1996.

[4] E. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *Proceedings of ACM CCS*, pages 132–145, 2004.

[5] Economist.com. The end of privacy. http://www.economist.com/displaystory.cfm?story_id=202103, April 1999.

[6] EPIC Report. Surfer beware: personal privacy and the internet. http://www.epic.org/reports/surfer-beware.html, June 1997.

[7] eWeek.com. Intel, AMD call for innovation on virtual platforms. http://www.eweek.com/article2/0,1895,1772065,00.asp, March 2005.

[8] eWeek.com. TPM hardware offers easier security. http://www.eweek.com/article2/0,1895,1847514,00.asp, August 2005.

[9] FIPS PUB 197. Advanced Encryption Standard (AES). National Institute of Standards and Technology, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, November 2001.

[10] T. Fukuhara and D. Singer. 15444-3 amendment 2, Motion JPEG2000. Motion JPEG2000 Version 2, MJP2 derived from ISO media file format. *ISO/IEC JTC 1/SC 29/WG1 N2780F*, January 2003.

[11] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In *Proceedings of the 19th Symposium on Operating System Principles (SOSP 2003)*, pages 193–206, October 2003.

[12] M. Hohmuth, H. Hartig, and J. S. Shapiro. Reducing TCB size by using trusted components - small kernels versus virtual machine monitors. In *Proceedings of the 11th ACM SIGOPS European Workshop*, September 2004.

[13] M. Kaplan. IBM Cryptolopes^{TM} - superdistribution and digital rights management. http://www.research.ibm.com/people/k/kaplan, December 1996.

[14] W. Li. Overview of fine granularity scalability in MPEG-4 video standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):301–317, March 2001.

[15] R. Mori and M. Kawahara. Superdistribution: The concept and the architecture. *Transaction of the IEICE*, E73(7):1133–1146, July 1990.

[16] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[17] J. Quittner. Invasion of privacy. *Time Magazine*, 150(8), August 1997.

[18] M. Rabbani and R. Joshi. An overview of the JPEG2000 still image compression standard. *Signal Processing: Image Communication*, 17(1):3–48, 2002.

[19] M. Reed, P. Syverson, and D. Goldschag. Anonymous connections and Onion Routing. *IEEE J. Selected Areas in Communications*, 16(4):482–494, May 1998.

[20] M. Reiter and A. Rubin. Crowds: anonymity for web transactions. *ACM Transactions on Information System Security*, 1(1):66–92, November 1998.

[21] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends. *IEEE Computer*, 38(5):39–47, May 2005.

[22] D. Taubman and M. Marcellin. *JPEG2000 Image Compression Fundamentals, Standard and Practice*. Kluwer Academic Publishers, 2000.

[23] Trusted Computing Group. Trusted Platform Module (TPM) specification. https://www.trustedcomputinggroup.org/groups/tpm, October 2003.

[24] Trusted Computing Group. Trusted Platform Module (TPM) FAQ. https://www.trustedcomputinggroup.org/groups/tpm/TPM_FAQ_2005.pdf, 2005.

[25] B. B. Zhu, M. B. Feng, and S. Li. An efficient key scheme for layered access control of MPEG-4 FGS video. In *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo*, pages 443–446, June 2004.